This extended cluster **23** is described in greater detail below, following a background description of FIG. **4**.

FIG. **4** shows a multiple cluster system **30** in which the back-end **31** includes MXE **1**, MXE **2**, . . . MXE**15**, MXE**16**, communicating with a central MXE **32**. Central MXE **32** then communicates with redundant front-ends **33**, which communicate with LNX controllers (LNX**1** and LNX**2**), which communicate to the real network channels **14** as described previously with respect to FIG. **1**.

For this extended configuration, it should be possible to have the single central node sitting inbetween two (or possibly more) front-ends and an array of back-ends. Using present technology, sixteen back-end machines could yield a combined subscriber capacity of, for example, 800,000, where the central node would be necessary to reduce the number of signaling links to the back-end, and for managing the subscriber lookup-tables.

The ability to expand a single cluster is limited by 1) the ability to configure additional channels (both real and virtual), and 2) the load capacity of the front end machine. Beyond those limitations, the present invention contemplates multiple clustering.

The following describes what is believed to be the most practical and efficient means for expanding into a double-cluster or other multiple-cluster arrangement. A second level of clustering begins by first setting up a duplicate of the single cluster that has already been described.

Once in place, the two independent clusters can be "meshed" such that each front-end machine, instead of being connected to three back-end machines, will now be connected to all six back-end machines. Each back-end machine will, however, only have 60 channels (for example) going to each front-end machine (rather than 120 available (for example) in stand-alone or single cluster mode). That is, the number of channels for each signaling link is reduced from 120 to 60, but the same traffic load is maintained as there are now two circuit switching controllers, one to each front end machine. Thus, each back-end machine will still have 120 channels, because a second LNX controller is added, thus permitting 60 channels coming from each front-end to each of these back-end controllers.

One might expect this to limit the system to only 60 channels. However, to begin with, this is unlikely because the network will distribute the traffic evenly when it selects a route to the MXE, and the channels will, therefore, be distributed evenly between the two front-end machines. There is, however, the possibility still that all channels from one front-end to a given back-end are occupied, while nearly all channels from the other front-end are available.

To handle this, the network channels for each front-end machine are divided into two LNX channel groups. Now the second group of channels for each cluster will be for the LNX on the other cluster as there are now two LNXs, each having two CPU boards. These CPU boards include the ability to be able to switchover to a standby LNX matrix. That is, the software that controls the LNX is designed, such that if a second LNX matrix board is configured, and if either the active board becomes faulty or the control link is disconnected, the MXE will detect this failure and begin to use the standby system. So two links are available, where the second link is going to the standby matrix on the other LNX.

FIG. **14** illustrates an example of this embodiment. The two LNX's are shown (one for each front end). Each LNX has an active board and a standby board. The active board is linked to the corresponding front end. That is, the first front end associated with the first LNX are linked together. The front end receives the active board link from the corresponding LNX at the LNX software module of the front end. A redundant module is also included in each front end to receive a redundant pair. This pair includes a link from the standby board of the opposite LNX and a link from the VLX module of the opposite front end. Thus, each front end receives three cables: 1) from the single active LNX board on the corresponding LNX, 2) from the redundant standby LNX board on the opposite LNX, and 3) from the VLX module of the opposite front end.

Ordinarily, the standby board of each LNX carries no traffic. Thus, each front end receives traffic from the active link of its corresponding LNX and from the VLX module of its opposite front end. But, if one of the front ends crashes, the virtual link will die with it and the standby board will then take over. In the end, the LNX ahead of the crashed front end can still control traffic simultaneously with the other LNX to the one operating front end. One LNX will feed the operational front end via its active link to the LNX software module at the front end and the other LNX will feed the front end via its standby link to the redundant LNX software module at the front end.

When the system is configured for redundancy, each of the two controllers has two links. Once redundancy configurations are imposed, the first controller has one link remaining, while the second controller has two links. This second link of the second controller is connected to a port on the other front-end machine that has a VLX module generating LNX signals. Then, to be able to switch calls to the second front-end machine, in order to avoid the condition described above where all channels on one side are occupied, a set of VLX channels is simply configured to do so.

Next, the selected MMI must be guaranteed to be physically connected to the LNX that the call came in on. By configuring the MMI channels such that they have an appropriate hunt group, that is, a hunt group that corresponds to the LNX that the MMI channel is physically connected to, and by hasing the front-end service, for example, add a prefix to the address information that gets sent to the back-end, then the service running on the back-end can select an MMI such that it will always be connected to the correct LNX. In effect this is a communication scheme to allow the front-end to tell the back-end which LNX the call has come in on.

The description above with respect to FIG. **4** includes an example of how a "three-tier" configuration could be created that would allow for such a large-scale cluster. There is, however, a short-coming with that model, namely, that the "central" node would not have the thread capacity to handle the required number of calls (the current MXE software uses a thread for each channel, as well as a thread for each call). And it is also questionable as to whether a single node would be able to handle the required call rate, i.e., whether a single call could afford to go through two intermediary service nodes. One solution, therefore, would be to create a central node that, instead of running a service, would simply do a circuit translation and route the call-processing message. If we describe the service-driven front-end machine as a circuit-switching node, then we could describe the central machine as being a form of packet-switching node.

This solution also opens up the possibility for having a remote connection from a front-end machine to the central node, called the circuit switch access node (CSA) of FIG. **3**. This would typically be a leased line, i.e., X.25, going from one site where a telephony network exchange and a front-end MXE would co-reside, going to another site where the central node and the array of back-end MXE's would co-reside.